Phases of the project:

1. Preparing images
2. calculate LBP(local binary pattern) features
3. calculate Histogram features
4. calculate euclidean distances
5. produce results

1. Preparing face images:

for training datasets i have taken 4 images for one person, and images of 3 persons have been collected. So, total 12 images. Then, i extracted faces and resizes them to 100x100 pixels. Images of each person have been stored in separate folder.

Similarly for 1 face image to be tested, the same preprocessing is done.

Accessing images in the program:

Two text files are created, one contains locations(file path) of all training images and other one contains path to test images.

2. Finding LBP(local binary pattern) from image:

the size of the image is 100x100 = 10,000 pixels.

Then next, the image is converted to gray scale image. Where each pixel has a value between 0-255, 0 for pure black and 255 for pure white.

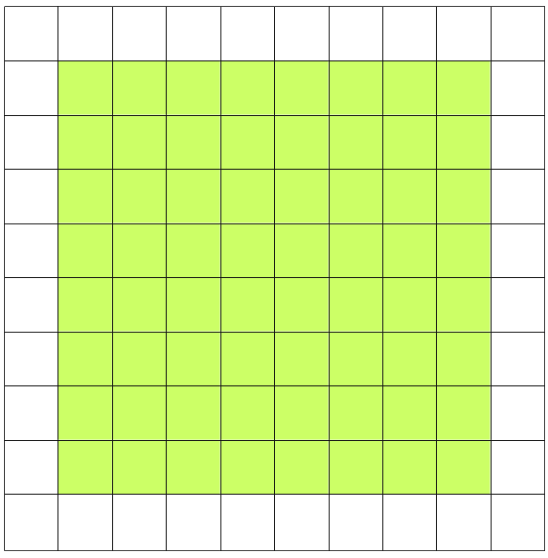
Next the image is partitioned into 10x10 blocks,

where each block contains 10x10 = 100 pixels,

then from each block LBP is calculated and stored in an array with associated block position as index.

calculating LBP for a block:

To calculate LBP for a block we have to select centers where each center has 8 neighboring pixel and then calculate LBP for each center.



10x10
A block

center

Discarded to be a center

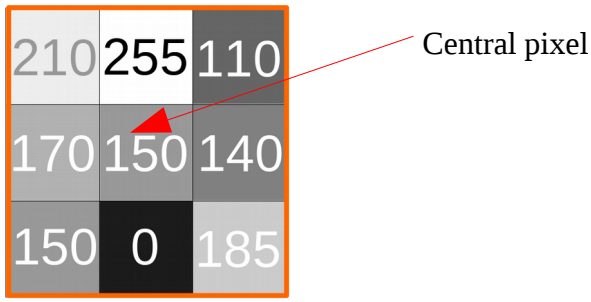
Total centers for one block = 8x8
= 64

Total centers for 1 image = 10x10x64
= 6400

i.e. 6,400 LBP features for one image

Calculating LBP for a center:

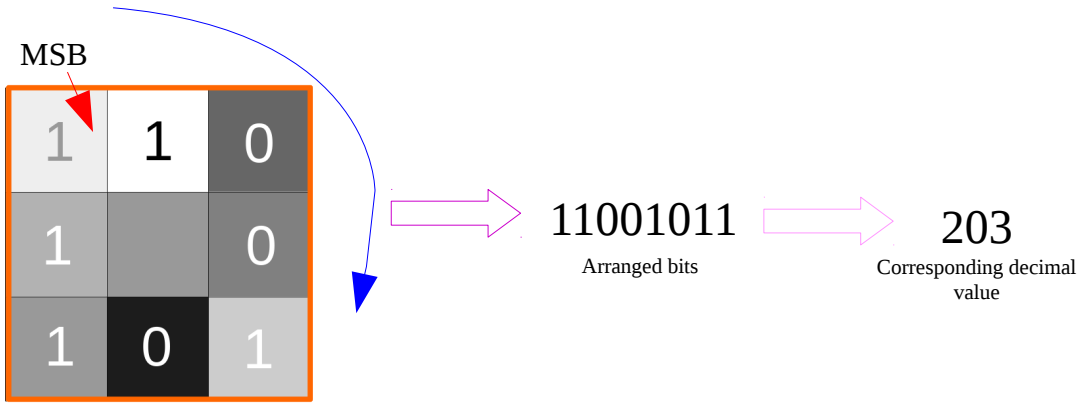
To illustrate how to find the LBP, the following picture with pixel intensities is considered.



Selected pixel as center to calculate LBP with 8 neighboring pixels

Now, compare each neighbor pixel with center, if the neighbor pixel has higher or equal intensity than central pixel then consider 1 for that position else consider 0.

then, we will have 8 numbers for 8 neighbors pixel, each one is either 0 or 1, so arrange those 8 bits by selecting any one bit as most significant bit(MSB) and then select other bits by moving either clockwise or anticlockwise. It is noted that, selecting bits can be in any order but the same order must be maintain for all centers in every block in every image. Then calculate the corresponding Decimal value of that 8 bit binary number and then store it in the array in corresponding center location as index.



After comparing pixel intensities

3. Calculating Histogram Data from LBP features:

After calculating LBP features of an image, we have now 6,400 LBP features from 10x10 = 100 blocks, now each block contains 64 LBP features which is 8x8 features in a block.

Now from that 8x8 features from each block we have to calculate histogram data.

In LBP features, every LBP value is in range from 0 to 255.

Next we select a range of intensity, say 5,

now we can make 255/5 = 51 +1 = 52 number of bars in histogram (from 0 to 51) each bar will represent frequency of intensity in a block in its corresponding intensity range.

Bar sl no	Intensity range	Bar sl no	Intensity range
0	0-4	26	130-134
1	5-9	27	135-139
2	10-14	28	140-144
3	15-19	29	145-149
4	20-24	30	150-154
5	25-29	31	155-159
6	30-34	32	160-164
7	34-39	33	165-169
8	40-44	34	170-174
9	45-49	35	175-179
10	50-54	36	180-184
11	55-59	37	185-189
12	60-64	38	190-194
13	65-69	39	195-199
14	70-74	40	200-204
15	75-79	41	205-209
16	80-84	42	210-214
17	85-89	43	215-219
18	90-94	44	220-224
19	95-99	45	225-229
20	100-104	46	230-234
21	105-109	47	235-239
22	110-114	48	240-244
23	115-119	49	245-249
24	120-124	50	250-254
25	125-129	51	255

It is to be noted that the last bar (i.e. sl no 51) got only 255

so, now there will be 52 histogram features for each block,
Total will be 52x10x10 = 5,200 features for an image
to calculate histogram data for an image the algorithm will be:

Step 1: initialize an 3d array (say bar) of size 10, 10, 52 and assign 0 to every location
(i.e. = bar[10][10][52])
Step 2: for each block_row_no in block in all rows do:
step 3: for each block_col_no in block_row_no do:
step 4: for each lbp_val in LBP features within the selected block do:
step 5: bar[block_row_no] [block_col_no] [lbp_val/5]++
step 6 end
step 7 end
step 8 end
step 9: done.

For example, the histogram data for the following 8x8 LBP features as follows:

200	100	0	5	255	45	78	120
99	255	189	156	222	243	24	64
32	46	245	56	245	26	2	1
57	37	120	153	12	75	47	46
74	36	88	246	24	67	24	16
183	166	76	210	88	46	53	84
36	63	37	3	78	35	45	84
36	99	34	253	255	0	25	45

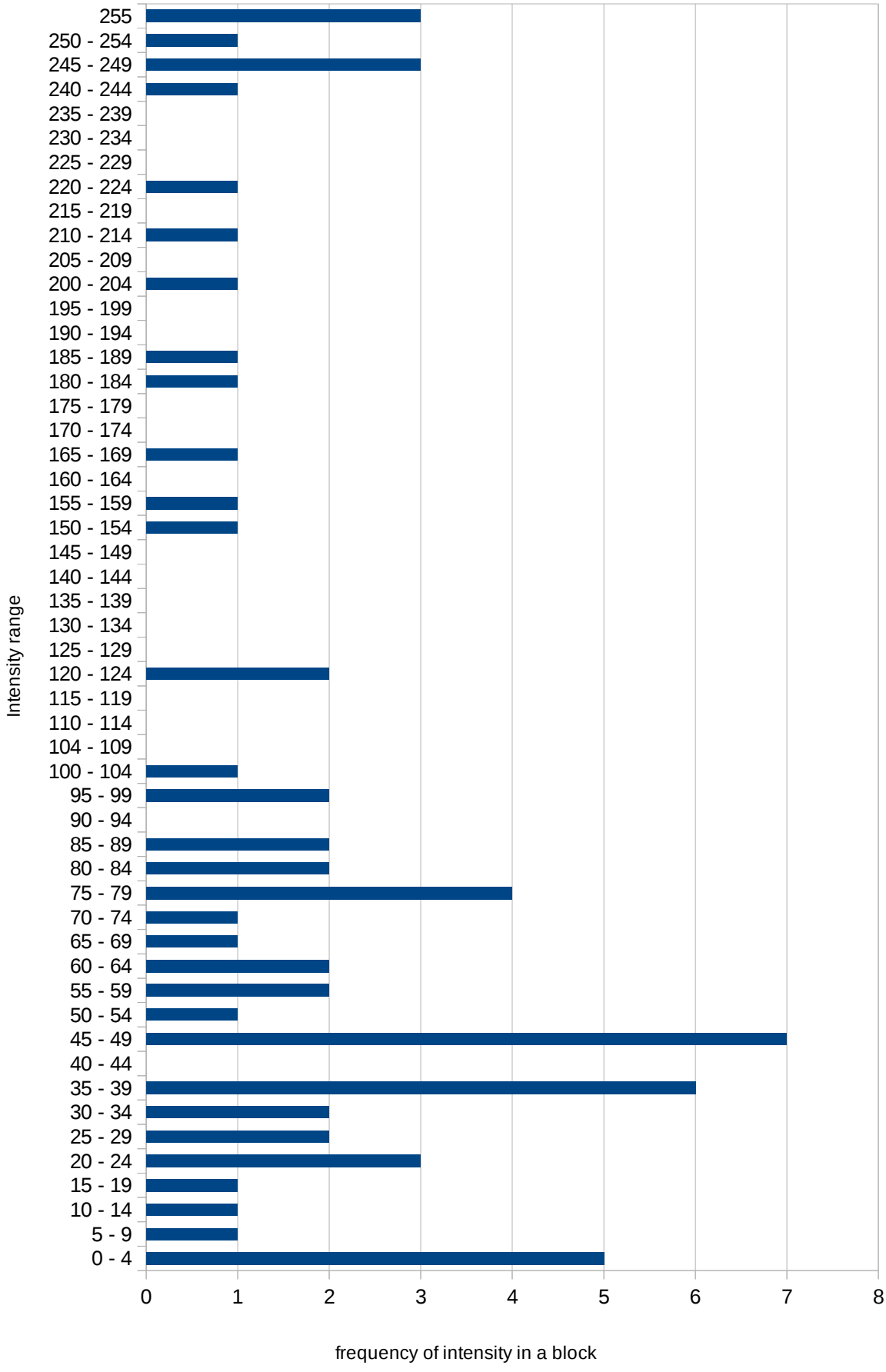


Fig: Histogram of the above LBP features

4. Calculating Euclidean distance:

the euclidean distance of any two value is computed as

eu_distance = $\sqrt{(val_1 - val_2)^2}$

for two 1D arrays A_1 and A_2 with n number of values is computed as:

eu_distance = $\sum_{i=1}^{n} \sqrt{(A_1[i] - A_2[i])^2}$

Now, we have Histogram data of images, contains 5,200 features per image. Now to find euclidean distance between histogram data of two images, the algorithm is:

step 1: initialize eu_distance = 0
step 2: for each block_row_no in block in all rows do:
step 3: for each block_col_no in block_row_no do:
step 4: for each h_val_1 in histogram features within the selected block of $image_1$ and h_val_2 in histogram features within the selected block of $image_2$ do:
step 5: eu_distance = eu_distance + sqrt(square($h\_val_1 - h\_val_2$))
step 6: end
step 7: end
step 8: end
step 9: done.

5. Producing Result:

The purpose of the system in this project is to identify a person from face image using already stored minimum number of face images.

The euclidean distance can now tell the difference of images, higher the euclidean distance lower the chance of match, and lower the euclidean distance higher the chance of match.

If we have n number of images stored and we want to identify a person face, then there will be n number of euclidean distances between image to be tested and stored images. Now the stored image with lowest euclidean distance will be the result, and may be the person to be identified. This is done when there is only one image stored per person.

But when there is more than one image stored per person, then euclidean distance between one person images with tested image will be sum of all euclidean distances between each images of that person with the tested image. Then the minimum euclidean distance will be the resulted person to be identified. The algorithm is as follows :

```
step 1: initialize min_eu = +inf, eu_distance = 0,resulted_person = person[0]
step 2: for each person p in person_images_database:
step 3:         for each image i in p do:
step 4:                 eu_distance = eu_distance + euclidean_distance(i,test_image)
step 5:         end
step 6:         if min_eu > eu_distance then
step 7:                 min_eu = eu_distance
step 8:                 resulted_person = p
step 9:         end
step 10:        eu_distance = 0
step 11: end
step 12: done.
```